НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
"___"_____2020 р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Спеціалізовані комп'ютерні системи»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Керований генератор псевдовипадкових двійкових векторів

Виконав:

студент IV курсу, групи КВ-62
(шифр групи)

Галицький Данііл Володимирович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник Проф. каф.СПіСКС, д.т.н., доцент Віталій РОМАНКЕВИЧ
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Ярослав Клятченко
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___»_____2020 р.

ЗАВДАННЯ
на дипломний проект студента
Галицького Данііла Володимировича

1.Тема проєкту Керований генератор псевдовипадкових двійкових векторів керівник Проф. каф.СПіСКС, д.т.н., доцент Віталій РОМАНКЕВИЧ затверджені наказом по університету від «25» травня 2020р №1181-С

2.Термін подання студентом проєкту 05.06.2020р

3.Вихідні дані до проєкту див. Технічне завдання.

4.Зміст пояснювальної записки

- Загальні відомості про ГПВЧ

- Аналіз існуючих рішень

- Опис розробленого генератора псевдовипадкових двійкових векторів

- Аналіз методів прискорення розповсюдження переносу

- Використання групового переносу для пришвидшення роботи розробленого ГПВЧ

- Математична модель залежності швидкодії схеми від кількості елементів схеми ГПВЧ з груповим переносом

- Опис програми моделюючої ГПВЧ з груповим переносом

- Результати тестування програми, моделюючої ГПВЧ з груповим переносом

5.Перелік графічного матеріалу(із зазначенням обов'язкових креслеників, плакатів, презентацій, тощо)

- Керований генератор псевдовипадкових двійкових послідовностей. Схема функціональна

- Генератор з групуванням по 4 елементи. Схема функціональна

- Генератор з групуванням по 3 елементи. Схема функціональна

- Структурна схема генератора

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7.Дата видачі завдання «1» листопада 2019р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	10.02.2020
2.	Розроблення та узгодження технічного завдання	13.03.2020
3.	Аналіз існуючих рішень	18.03.2020
4.	Підготовка матеріалів розділів дипломного проекту	10.04.2020
5.	Підготовка звіту дипломного проекту	10.05.2020
6.	Передзахист дипломного проекту	20.05.2020

Студент _____ Даніїл Галицький
(підпис) (Ім'я та ПРІЗВИЩЕ)

Керівник проекту _____ Віталій Романкевич
(підпис) (Ім'я та ПРІЗВИЩЕ

*Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

Мета дипломного проекту-створення керованого генератора псевдовипадкових двійкових векторів.

Для реалізації проведено аналіз існуючих схем генераторів та способів покращення швидкодії подібних схем. У результаті роботи було створено схему керованого генератора псевдовипадкових двійкових векторів та математичну модель, описуючу покращення швидкодії генератора при застосуванні додаткової схеми групового переноса при збереженні вихідної послідовності генератора. Результати дипломної роботи можуть бути використанні при апаратній реалізації керованого генератора псевдовипадкових двійкових векторів.

Ключові слова: генератор, псевдовипадкові двійкові вектори, груповий перенос.

SUMMARY

The purpose of the diploma project is to create a controlled generator of pseudo-random binary vectors.

For implementation, an analysis of existing generator circuits and ways to improve the performance of such circuits was performed. As a result of the work the was created the scheme of the controlled generator of pseudo-random binary vectors and a mathematical model describing the improvement of the generator speed when using an additional group transfer scheme while saving the original sequence of the generator. The results of the thesis can be used to create a hardware implementation of a controlled generator of pseudo-random binary vectors.

Keywords: generator, pseudo-random binary vectors, group transfer.

По	Фор	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кіль кіст	№	Примітки
	А4	ІАЛЦ.045490.002 ТЗ	Керований генератор	4		
			псевдовипадкових			
			двійкових векторів			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Керований генератор	2		
			псевдовипадкових			
			двійкових векторів			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045490.004 ПЗ	Керований генератор	51		
			псевдовипадкових			
			двійкових векторів			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Е2	Керований генератор	4		
			псевдовипадкових			
			двійкових векторів			
			Копії графічних матеріалів			
	А4	ІАЛЦ.045490.005 Е2	Керований генератор	3		
			псевдовипадкових			
			двійкових векторів			
			Лістинг програми, що моделює розроблений генератор			

Змін.	Арк.	№ докум.	Підпис	Дата

ІАЛЦ.045490.001 ОА

Розробив	Галицький Д. В.	Керований генератор псевдовипадкових двійкових векторів	Літ. / Аркуш 1 / Аркушів 2
Перевірив	Романкевич В.О.		
Консульт.			КПІ ім. Ігоря Сікорського Кафедра СПіСКС Група КВ-62
Н. контроль	Клятченко Я.М.		
Зав. каф.	Романкевич В.О.	Опис альбому	

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Керований генератор псевдовипадкових двійкових векторів».

Галузь застосування: моделювання, криптографія та інформаційна безпека,.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є розробка схеми керованого генератора псевдовипадкових двійкових векторів.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до продукту, що розробляється

- можливість керування керованим генератором псевдовипадкових двійкових векторів

- успішне проходження керованим генератором псевдовипадкових двійкових векторів усіх поліноміальних статистичних тестів
- максимально можлива швидкодія схеми генератора псевдовипадкових двійкових векторів

5.2 Вимоги до програмного забезпечення користувача

- Операційна система Windows
- 1гб ОЗУ
- Процессор з таковою частотою 1,60 GHz

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	10.02.2020
2.	Розроблення та узгодження технічного завдання	13.03.2020
3.	Аналіз існуючих рішень	18.03.2020
4.	Підготовка матеріалів розділів дипломного проекту	10.04.2020
5.	Підготовка звіту дипломного проекту	10.05.2020
6.	Передзахист дипломного проекту	20.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045492.004 ПЗ	Керований генератор	51		
			псевдовипадкових			
			двійкових векторів			
			Пояснювальна записка			
	А4	ІАЛЦ.045492.005 Е1	Керований генератор	1		
			псевдовипадкових			
			двійкових векторів			
			Схема структурна			
	А4	ІАЛЦ.045492.006 Е2	Генератор	1		
			з групуванням			
			по 4 елемента			
			Схема функціональна			
	А4	ІАЛЦ.045492.007 Е2	Генератор	1		
			з групуванням			
			по 3 елемента			
			Схема функціональна			

Змін.	Арк.	№ докум.	Підпис	Дата	***ІАЛЦ.045492.003 ТП***		
Розробив		Галицький Д.В			Генератор псевдовипадкових двійкових векторів **Відомість технічного проекту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Романкевич В.О					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ КВ-62	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В. О					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А1	ІАЛЦ.045492.008 Е2	Керований генератор	1		
			псевдовипадкових			
			Лістинг програми, що			
			моделює розроблений			
			генератор			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045492.003 ТП	Арк. 2

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ГПВЧ-генератор псевдо випадкових чисел

ГВБ-генератор випадкових біт

ГПВБ-генератор псевдо випадкових біт

FCSR-feedback with carry shift register, Регістр зсуву зі зворотним зв'язком по перенесенню

LFSR- linear feedback shift register, Регістр зсуву з лінійної зворотним зв'язком

ВСТУП

Випадкові числа використовуються давно і повсюдно. Перерахуємо деякі області їх застосування:

1. Соціологічні та наукові дослідження. Підготовка випадкових вибірок при зборі даних, опитуванні думок або в дослідженні фізичних явищ з випадковим вибором результатів експериментів.

2. Моделювання. У комп'ютерному моделюванні фізичних явищ. Крім того, математичне моделювання використовує випадкові числа як один з інструментів чисельного аналізу.

3. Криптографія та інформаційна безпека. Випадкові числа можуть використовуватися в тестуванні коректності або ефективності алгоритмів і програм. Багато алгоритмів використовують генерацію псевдовипадкових чисел для вирішеіня прикладних завдань (наприклад, криптографічні алгоритми шифрування, генерація унікальних ідентифікаторів та ін.).

4. Прийняття рішень в автоматизованих експертних системах. Використання випадкових чисел є частиною стратегій прийняття рішень. Наприклад, для неупередженості вибору екзаменаційного білета студентом на іспиті. Випадковість також використовується в теорії матричних ігор.

5. Оптимізація функціональних залежностей. Деякі математичні методи оптимізації використовують стохастичні методи для пошуку екстремумів функцій.

6. Розваги та ігри. Випадковість в іграх має значну роль. У комп'ютерних або настільних іграх випадковість допомагає урізноманітнити ігровий процес.

Необхідність використовувати випадкові числа в науковій роботі виникла давно.

Спочатку для цієї мети використовувалася урна з кулями, з якої навмання витягали кулю з цифрою. Пізніше були побудовані механічні генератори випадкових чисел. З появою електронних схем з'явилися і електронні генератори випадкових чисел. Один з перших таких генераторів, запропонований А.М.Тьюрингом, використовував резисторний генератор шуму для отримання

20 випадкових біт, що надходили на суматор . Однак генератори випадкових чисел не завжди давали «хороші» результати (поняття «хороших» випадкових чисел буде розглянуто далі). Крім того, апаратні генератори випадкових чисел нерідко давали збої. Таблиці ж «хороших» випадкових чисел, обчислених заздалегідь, були вкрай незручні у використанні у зв'язку з обмеженістю комп'ютерної пам'яті.

У 90-і роки 20 століття зростання можливостей комп'ютера, збільшення щільності запису на магнітних і оптичних носіях дозволило скласти досить великі таблиці випадково згенерованих бітів. Так наприклад,

Джордж Маршал створив великий каталог таблиць випадкових чисел обсягом 650 мегабайт, що дозволило розмістити його на оптичному диску.

1.Загальні відомості про ГПВЧ

1.1. Основні поняття

Двійковим (булевих) n-мірним вектором називають набір з n чисел (b1, b2, ..., bn),, кожне з яких може приймати тільки значення в двійковій системі числення - 0 або 1.

Двійковий вектор називають обратним (інвертованим) до, якщо він утворений з заміною всіх нулів одиницями, а одиниць - нулями. Наприклад, якщо = (0, 1, 1, 1, 0, 1), то = (1, 0, 0, 0, 1, 0).

Якщо в запису двійкового вектора довжиною n усунути коми, то його можна розглядати як двійковий запис деякого цілого числа. Найменшим таким числом є нуль. Йому відповідає вектор = (0, ..., 0). Найбільшим є число 2n - 1, якому відповідав би вектор = (1, ..., 1). Отже, за допомогою повного набору двійкових векторів довжиною n можна записати все 2n цілих чисел з відрізка [0, 2n - 1]. Такі числа називають порядковими номерами векторів. Їх використовують як в двійковому вигляді, так і в десятковій системі числення. Двійковий запис довільного числа можна розглядати як послідовність двійкових чисел.

Послідовність називається випадковою, якщо відтворити її, знаючи алгоритм і всі вихідні дані не представляється можливим (двічі запустивши генератор в тих же умовах ми отримаємо різні послідовності). Але комп'ютерні системи детерміновані, тобто вони можуть мати лише кінцеву кількість станів. Це призводить до того, що послідовності які генеруються ними і будуть періодичні - такі послідовності називаються псевдовипадковими.

Зважаючи на вище зазначене отримано двійкову послідовність довжиною n можна розглядати як двійковий запис випадкового або псевдовипадкового двійкового вектору. Вектор буде випадковим або псевдовипадковим яким чином була здійснена генерація вектору. Генерація може бути здійснена двома основними способами :

- шляхом використання пристроїв в основу роботи яких покладено використання різноманітних фізичних явищ таких як дробовий шум, радіоактивний розпад та інші ;
- використання програмного забезпечення або цифрових апаратів які реалізують детерміновані алгоритми знаходження всіх чисел ,що складають двійковий запис вектору.

В залежності від методу генерації двійкових векторів ,алгоритмів роботи генераторів (програмних чи апаратних) ми будемо отримувати або випадкові або псевдовипадкові двійкові вектори.

Спочатку наведемо визначення основних понять теорії генерації випадкових чисел.

Визначення 1.1. Випадкове число - число, що представляє собою реалізацію випадкової величини .

Визначення 1.2. Детермінований алгоритм - алгоритм, який повертає ті ж вихідні значення при тих же вхідних значеннях.

Визначення 1.3. Псевдовипадкове число - число, отримане детермінованим алгоритмом, що використовується в якості випадкового числа.

Визначення 1.4. Фізичне випадкове число (істинно випадкове) – випадкове число, отримане на основі деякого фізичного явища.

Як правило, генерація випадкового числа складається з двох етапів:

1. генерація нормалізованого випадкового числа (тобто рівномірно розподіленого від 0 до 1);
2. перетворення нормалізованих випадкових чисел у випадкові числа, які розподілені по заданому закону розподілу або в необхідному інтервалі.

Визначення 1.5. Генератор випадкових біт (ГВБ) - це пристрій або алгоритм, який видає послідовність статистично незалежних і незміщених біт (тобто підкоряються закону розподілу).

Зауваження. Генератор випадкових біт може бути використаний для генерації рівномірно розподілених випадкових чисел. Наприклад, випадкове ціле число в інтервалі *[0; n]* може бути отримано з генератора випадкових біт довжини *⌈lg n⌉ + 1* шляхом конвертації її у відповідну систему числення. Якщо отримане в результаті ціле число перевершує n, то його можна відкинути і згенерувати ще одну послідовність біт.

Тому далі ми будемо використовувати термін генератор випадкових чисел нарівні з терміном генератор випадкових біт.

Визначення 1.6. Генератором псевдовипадкових біт (детермінованим ГПСБ) будемо називати детермінований алгоритм , який отримує на вхід двійкову послідовність довжини k і видає на виході двійкову послідовність довжини $l \gg k$ (l значно більше k), яка« виглядає випадковою ». Вхідне значення ГПВБ називається початковим вектором (також називають ініціалізованим вектором і позначають IV), а вихід називається псевдовипадковою послідовністю біт.

Пояснимо поняття «виглядає випадковою». Зрозуміло, що послідовність, згенерована детермінованим алгоритмом, не є випадковою. Однак мета алгоритму в тому, щоб взяти деяку маленьку послідовність істинно випадкових чисел і використовувати її для генерації довгої послідовності, яка не відрізняється від істинно випадкової послідовності чисел тієї ж довжини. Переконатися в тому, що послідовність чисел випадкова (або не випадкова) можна або за допомогою статистичних тестів, які виявлятимуть специфічні особливості випадкових послідовностей, або аналітико-обчислювальними методами.

Визначення 1.7. Кажуть, що ГПВБ проходить все поліноміальні за часом ймовірнісні тести на статистичну випадковість, якщо не існує поліноміального за часом імовірнісного алгоритму, який би міг коректно відрізнити вихідну послідовність генератора від істинно випадкової послідовності тієї ж довжини з ймовірністю, що перевищує 1/2.

Визначення 1.8. Кажуть, що ГПВБ успішно проходить тест на наступний біт, якщо не існує поліноміального за часом алгоритму, який може по вхідним *l* бітів послідовності s передбачити *(l + 1)* -й біт s з імовірністю, що перевищує 1/2.

Більш формально, ГПВБ проходить тест на наступний біт, якщо для будь-якого $i \in \mathbb{N}$ і будь-якого ймовірного поліноміального за часом алгоритму $A:\{0,1\} \rightarrow \{0,1\}$, виконується наступна нерівність:

$$\left| P(A(s_i^{i-1} - \frac{1}{2} \right| < O(v(n)),$$

де *O(v(n))*- позначення функції, спадної швидше, ніж зворотний поліном ступеня n.

Незважаючи на те, що визначення 2.7 накладає більш суворі умови

на ГПСЧ, ніж визначення 2.8, можна довести, що ці визначення еквівалентні.

Затвердження 1.1. (Універсальність тесту на наступний біт) ГПСБ проходить тест на наступний біт тоді і тільки тоді, коли він проходить всі поліноміальні статистичні тести.

Доказ цього твердження отримав Ендрю Яо в 1982 році.

1.2. Види генераторів випадкових чисел

Генератори випадкових чисел за способом отримання чисел розподіляються на:

1. апаратні;

2. табличні;

3. алгоритмічні.

Табличні генератори в якості джерела випадкових чисел використовують заздалегідь підготовлені таблиці, що містять перевірені некорельовані числа і не є генераторами в суворому розумінні цього поняття. Недоліки такого способу очевидні: використання зовнішнього ресурсу, обмеженість послідовності, зумовленість значень.

Апаратні генератори (істинно) випадкових послідовностей повинні володіти джерелом ентропії. Розробка генераторів, що використовують джерела ентропії, генеруючих не корельовані і статистично незалежні числа - досить складне завдання.

Крім того, для більшості криптографічних додатків такий ГПВЧ не повинен бути предметом вивчення і впливів іншої сторони.

Алгоритмічний генератор є комбінацією фізичного генератора і детермінованого алгоритму. Такий генератор використовує обмежений набір даних, отриманий з виходу фізичного генератора для створення довгої послідовності чисел щляхом перетворення вихідних чисел.

2. Аналіз існуючих рішень

2.1 Генератор випадкових чисел Г. Гібсона

У своїй книзі ‘Tronix book 2’ Г.Гібсон наводить наступний приклад генератора випадкових чисел:

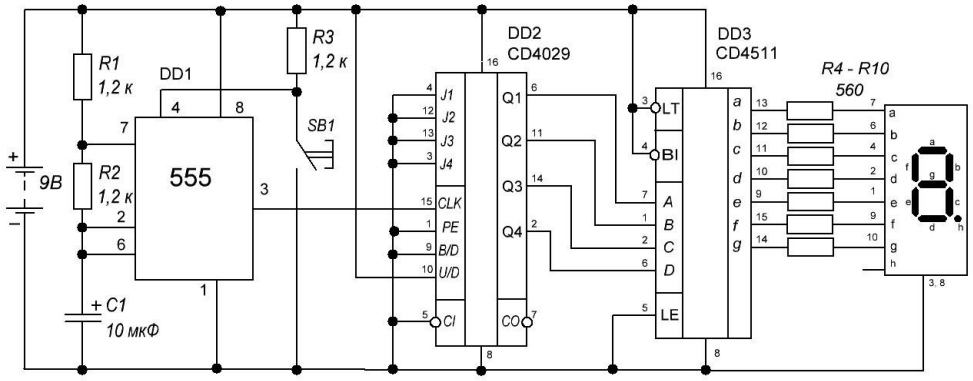

Рисунок 2.1 Схема генератора випадкових двійкових чисел, розробленого Г.Гібсоном

Дану схему можна умовно поділити на чотири елементи, що послідовно виконують певні функції, а після вихід одного елемента перенаправляється на вхід наступного. Список цих елементів і послідовність їх роботи виглядає так:

1. Генератор імпульсів
2. Лічильник
3. Дешифратор
4. Семисегментний індикатор

2.1.1 Генератор імпульсів

Як генератор імпульсів виступає популярний таймер 555. На виході таймер видає серію імпульсів, яка буде «підштовхувати» лічильник і змушувати його додавати значення.

Основний таймер 555 отримав свою назву через те, що є три внутрішніх резистора 5 кОм, які він використовує для генерації опорних напруг двох компараторів. ІС таймера 555 - це дуже дешевий, популярний і корисний пристрій точного хронування, який може діяти як простий таймер для

Генерації одиночних імпульсів або тривалих затримок, або як генератор релаксації, що генерує ланцюжок стабілізованих сигналів з різними робочими циклами від 50 до 100%.

Мікросхема таймера 555 є надзвичайно надійним і стабільним 8-контактним пристроєм, який може працювати як дуже точний моностабільний, бістабільний або нестабільний мультивибратор для створення різних застосувань, таких як таймери одноразового спрацьовування або затримки, генерація імпульсів, світлодіодні і лампові проблискові маячки, сигналізація і генерація тону, логічний годинник, частотне розділення, джерела живлення і перетворювачі тощо, фактично будь-яка схема, яка вимагає певної форми контролю часу, оскільки список нескінченний.

Єдина мікросхема 555 Timer у своїй базовій формі являє собою біполярний 8-контактний міні-пристрій з подвійним входом у лінію (DIP), що складається з приблизно 25 транзисторів, 2 діодів і близько 16 резисторів, зкомпонованих для формування двох компараторів, тригера і вихідного каскаду високого струму, як показано нижче. Поряд з таймером 555 є також генератор таймера NE556, який об'єднує дві окремі моделі 555 в одному 14-контактному DIP-корпусі і малопотужні CMOS-версії одного таймера 555, такі як 7555 і LMC555, в яких замість цього використовуються транзистори MOSFET.

Спрощена «блок-схема», що представляє внутрішню схему таймера 555, наводиться на рисунку 2.2 з коротким поясненням кожного з його сполучних штифтів, щоб допомогти забезпечити більш чітке розуміння того, як він працює.

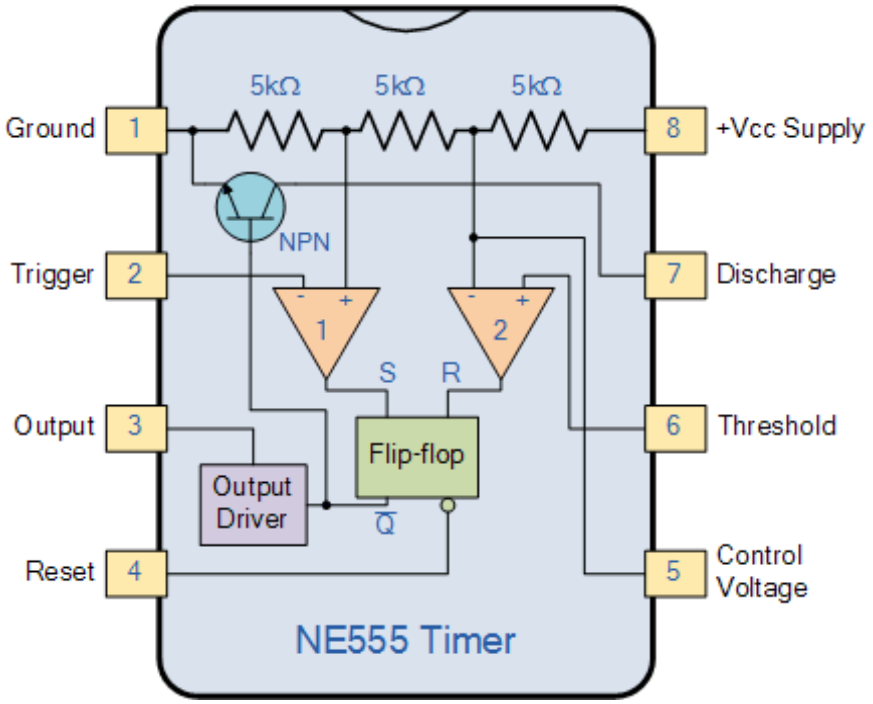

Рисунок 2.2 Спрощена блок-схема таймера 555

• Контакт 1. - Заземлення, заземлюючий штифт підключає 555 таймер до негативної (0 В) подачі.

• Контакт 2. - Тригер, від'ємний вхід до компаратора № 1. Від'ємний імпульс на цьому штифті "встановлює" внутрішній фліп-флоп, коли напруга опускається нижче 1 / 3Bcc, що призводить до переходу виходу з "НИЗЬКОГО" на "ВИСОКИЙ" стан.

• Контакт 3. - Вихідний висновок. Вихідний штифт може керувати будь-яким ланцюгом TTL і здатний подавати або пропускати до 200 мА струму при вихідній напрузі, рівній приблизно Vcc - 1,5 В, тому невеликі колонки, світлодіоди або двигуни можна підключити безпосередньо до виходу.

• Контакт 4. - Скидання. Цей штифт використовується для "скидання" внутрішнього фліп-флоп, керуючи станом виходу, контакт 3. Це активно-низький вхід і зазвичай підключений до логічного рівня "1", в іншому випадку використовується для запобігання будь-якого небажаного скидання виводу.

• Контакт 5. - Управляюча напруга. Цей штифт регулює час 555, переосмислюючи рівень 2 / 3Vcc мережі дільника напруги.

Застосовуючи напругу до цього контакту, ширину вихідного сигналу можна змінювати незалежно від мережі синхросигналу RC. Якщо не використовується, він підключається до землі за допомогою конденсатора 10nF для усунення будь-якого шуму.

• Контакт 6. - Поріг, позитивний вхід до компаратора № 2. Цей штифт використовується для скидання фліп-флопа, коли напруга, що подається на нього, перевищує 2 / 3Vcc, що призводить до переходу виходу з "ВИСОКОГО" в стан "НИЗЬКОГО". Цей штифт підключається безпосередньо до схеми синхронізації RC.

• Контакт 7. - Розряд, розрядний штифт підключається безпосередньо до колектора внутрішнього транзистора NPN, який використовується для «розрядки» конденсатора синхронізації на землю, коли вихід на штифт 3 перемикається «НИЗЬКО».

• Контакт 8. - Блок живлення + Vcc. Це контактний блок живлення, а таймери загального призначення TTL 555 - від 4,5 до 15 В.

У схемі генератора псевдовипадкових чисел, запропонованій Г.Гібсоном

підключення таймера до схеми виглядає так:

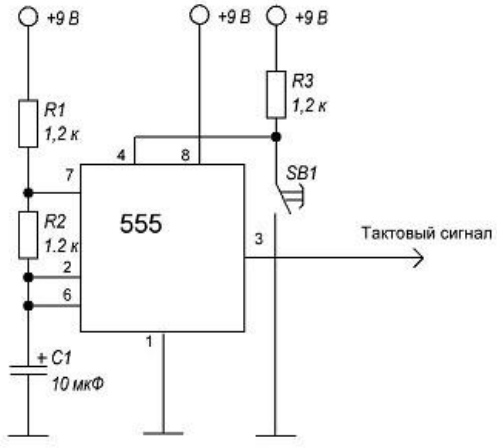

Рисунок 2.3 Підключення таймера 555 у схему генератора випадкових чисел Г.Гібсона

Режим роботи таймера задається трьома елементами: резисторами R1 і R2 та конденсатором C1.

2.1.2 Лічільник

Г.Гібсон використовує елемент cd4029n в якості лічильника.

У схемі генератора псевдовипадкових чисел, запропонованій Г.Гібсоном

підключення таймера до схеми виглядає так:

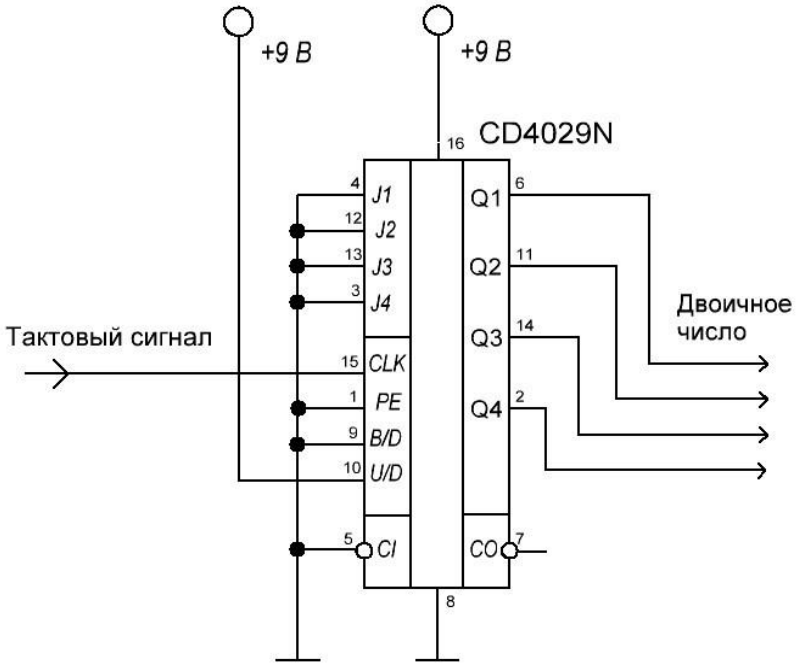

Рисунок 2.3 Підключення лічильника CD4029N у схему генератора випадкових чисел Г.Гібсона

Цифра виводиться як двійкове число на виходах $Q_1 \dots Q_N$

За сигналом на 15-му виведенні («Тактовий сигнал») лічильник додає одиницю.

У десятковому режимі після цифри 9 лічильник скидається і починає знову з 0.

Вхід U/D (напрямок рахунка) підключений до «плюса» живлення, щоб

рахунок йшов по зростаючій.

Виходи PE (дозвіл попередньо встановлений), j_1 ... j_N(входи даних для попереднього встановлення лічильника), B/D (двійковий режим рахунка), CI і CO (вхід і вихід перенесення), все підключене до «землі».

2.1.3 Дешифратор

В якості дешифратора в схемі виступає елемент 4511:

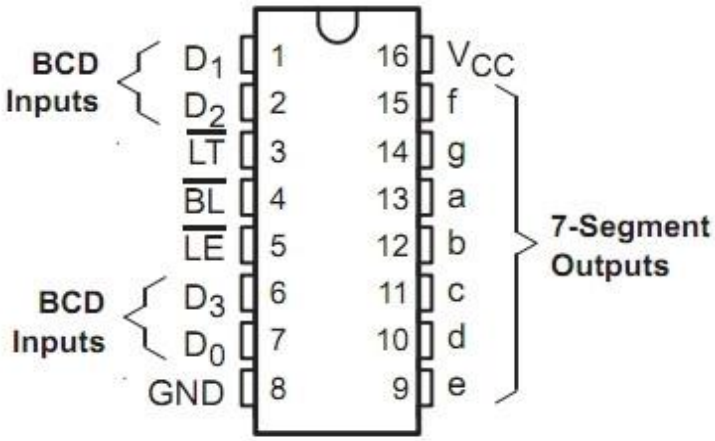

Рисунок 2.4 Схема елемента 4511(дешифратора)

Задача дешифратора - отримати на свої входи згенеровану попередніми двома елемента послідовність біт і поставити їй у відповідність набір сегментів на об'єкті типу «стандартний семисегментний індикатор».

Підключення дешифратора до схеми виглядає так:

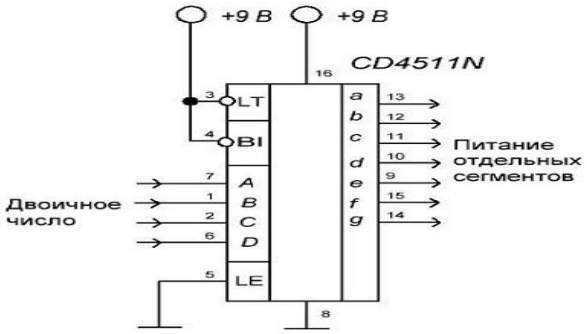

Рисунок 2.5 Підключення дешифратора до схеми генератора випадкових чисел Г.Гібсона

2.1.4 Семисегментний індикатор

В якості виводу схеми Г.Гібсон використовує семисегментний індикатор

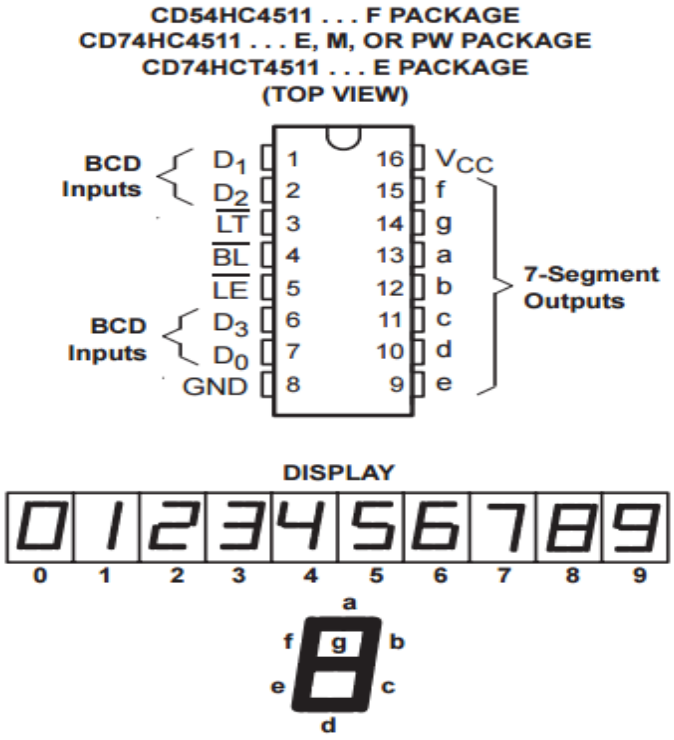

Рисунок 2.6 Семисегментний індикатор

Кожен із світлодіодів підключається через струмообмежуючий резистор. У документації семисегментного індикатора є спеціальна картинка, що зазначає варіант підключення через один резистор як невірний.

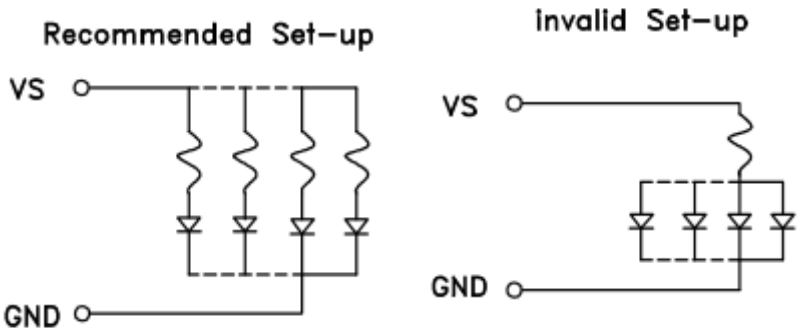

Рисунок 2.7 Підключення семисегментоного індикатора у схему генератора псевдовипадкових чисел Г.Гібсона

2.1.5 Аналіз схеми генератора випадкових чисел Г.Гібсона

Перша схема - генератор імпульсів - на виході має меандр, тобто прямокутний сигнал. Частота цього меандру задає швидкість роботи лічильника. Лічильник, у свою чергу, «розмовляє» з дешифратором на мові двійкового коду. А той, у свою чергу, передає дані в уже зручному для семисегментного індикатора вигляді.

Простота і логічність цієї схеми обумовлена тим, що мікросхеми в ній підібрані так, щоб вони найбільш легко стикувалися між собою.

2.2 Генератор псевдовипадкових двійкових послідовностей, наведений у книзі П. Хоровица та У.Хилла «Мистецтво схемотехніки»

В якості генератора псевдовипадкових послідовностей використовується керований регістр зсуву.

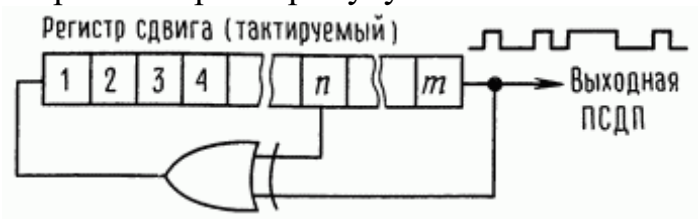

Рисунок 2.8 Керований регістр зсуву

Якщо кілька тригерів з'єднати так, що вихід Q кожного попереднього тригера буде управляти D-входом подальшого, а всі тактові входи будуть збуджуватися одночасно, то вийде схема, яку називають «регістр зсуву». По кожному тактовому імпульсу комбінація «нулів» і «одиниць» в регістрі буде зрушуватися вправо, а зліва через D-вхід першого тригера буде вводитися нова інформація.

Як і в усіх тригерних схемах, інформація на лівому вході, присутня безпосередньо перед виникненням тактового імпульсу, буде введена в регістр, і на виході буде звичайна затримка поширення. Таким чином, регістри можна об'єднати каскадно, не чекаючи виникнення режиму логічних гонок. Регістри зсуву широко використовуються для перетворення даних з паралельною форми (*n* біт надходить одночасно по *n* окремих ліній) в послідовну

(біти один за іншим передаються з інформаційної лінії) і навпаки. Вони також застосовуються в якості запам'ятовуючих пристроїв, особливо в тих випадках, коли дані зчитуються і записуються завжди однаковим чином. Регістри зсуву, як і лічильник, і фіксатори, представлені великим числом різноманітних модифікацій.

4-розрядні і 8-розрядні регістри є стандартними. Випускаються також регістри та з великим об'ємом (64 біта і більше). Існують навіть регістри зі змінною довжиною (наприклад, схема 4557 може змінювати свою довжину від 1 до 64 біт за допомогою входу управління).

Зазвичай регістри зсуву є поодинокими, однак випускаються також здвоєні та зчетверені регістри. Більшість регістрів зсуву виробляють зсув тільки вправо, але існують і регістри зі зсувом в обох напрямках, такі як 194 і 323, які мають вхід «напрямок»).

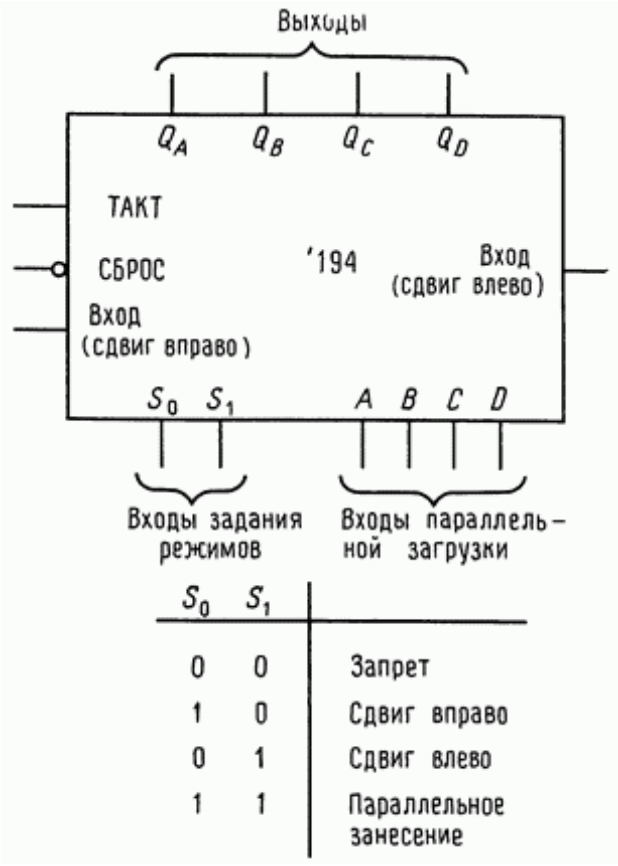

Рисунок 2.9 Чотирьохрозрядний реверсивний регістр зсуву

Невеликі регістри зсуву можуть виробляти паралельне введення і виведення, і зазвичай це роблять, наприклад, схема 395 є 4-розрядної регістром зсуву з паралельним введенням і висновком (PI / PO) з виходом на 3 стан. Великі регістри можуть здійснювати тільки послідовне введення і виведення, тобто допускається тільки введення в перший тригер або виведення з останнього.

У деяких випадках виводяться кілька проміжних виходів. Єдиний спосіб розмістити як паралельне введення, так і паралельний висновок в одному малому корпусі - це використовувати одні і ті ж контакти в якості входів і виходів. Так, наприклад, схема 299 являє собою 6-розрядний регістр паралельного введення / виведення (PI / PO) в контактному корпусі. Деякі зсувні регістри включають засувки (фіксатори) на вході або виході, так що зрушення може відбуватися поки дані завантажуються або вивантажуються.

Так само як і у лічильників, паралельне ЗАВАНТАЖЕННЯ і ОЧИЩЕННЯ можуть бути або синхронними, або асинхронними, наприклад схема 323 подібна до схеми 299, але з синхронним очищенням.

У схемі простого ГПВЧ, запропонованого у книзі П. Хоровица та У.Хилла “Мистецтво схемотехніки”, регістр зсуву довжини m працює від тактових імпульсів с частотою f_0.

Вхідна послідовність формується за допомогою вентиля “виключаюче або” , на вхід якого надходять сигнали від n-го і останнього розрядів регістра зсуву.

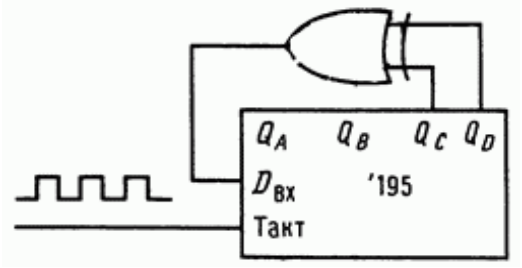

Рисунок 2.10 Регістр зсуву с вентилем “виключне або”

Така схема проходить через деяку кількість станів (сукупність станів регістра зсуву після кожного тактового імпульсу), які після k тактів починають повторюватися, тобто послідовність станів є циклічною з періодом K.

Максимальне число можливих станів m-розрядного регістра дорівнює $K = 2^m$, тобто дорівнює кількості m-бітових двійкових комбінацій. Однак стан "все нулі" є "тупиком" для цієї схеми, оскільки на виході вентиля "виключаюче або" з'являється 0, який знову надходить на вхід схеми. Таким чином, послідовність максимальної довжини, яку може сформувати дана схема, містить $2^m - 1$ біт. Виявляється, що таку послідовність максимальної довжини можна отримати тільки при правильному виборі m і n, причому отримана послідовність буде псевдовипадковою. (Критерієм максимальної довжини є неприводимість і примітивність многочлена $1 + x^m + x^n$ над полем Галуа). Як приклад розглянемо розрядний регістр зсуву зі зворотним зв'язком, показаний на рис. Починаючи зі стану 1111 (можна було б почати з будь-якого іншого стану, за винятком 0000), можна записати стани в порядку їх слідування:

1111 0100 1011

0111 0010 0101

0011 1001 1010

0001 1100 1101

1000 0110 1110

Стани можна записати як 4-разрядні числа $Q_a Q_b Q_c Q_d$. Тут 15 різних станів, потім вони повторюються знову.

3. Опис розробленого генератора псевдовипадкових двійкових векторів

Після детально аналізу існучих рішень, було розроблено керований генератор псевдовипадкових двійкових векторів із змінною ймовірністю, побудований на основі керованого регістра зсуву.

Загальний принцип роботи схеми полягає в організації зсуву двійкового коду в вихідному регістрі під управлінням сигналів від задаючого (опорного, керуючого) ГПВЧ, що формує рівноймовірні багаторозрядні набори. На рисунку 2.1 наведена узагальнена структура генератора псевдовипадкових двійкових послідовностей.

Вихідний N-бітний регістр Rg може бути реалізований на основі використання елементів затримки (наприклад, у вигляді тактуємих тригерів D-типу). Три логічні схеми SS_1, SS_2, SS_3 слугують для організації керованого зсуву в регістрі Rg: схема SS_1 містить елементи формування сигналів на інформаційних входах Rg, схема SS_2 організовує зв'язок бітів регістра Rg при керованому зсуві (іншими словами, це ланцюг поширення перенесення), схема SS_3 реалізує функціонал формування синхросигналів зсуву. За рахунок схем перенесення зрушення в регістрі Rg носить кільцевої характер. Якщо ЕП регістра Rg позначити $T_1, T_2 \dots T_N$, виходи схеми $SS_3 - C_{1,}, C_2 \dots C_n$, схеми $SS_1 - a_{1,}\ a_2 \dots a_N$ і виходи ГПСЧ - $g_1, g_{2\dots} g_N$, то для схеми перенесення SS_2 кільцевий характер з'єднання бітів регістра Rg полягає в тому, що виходи $b_1, b_2 \dots b_N$ залежать від перенесення із сусіднього біта, тобто

$$b_2 = f(b_1), b_3 = f(b_3) \dots b_N = f(b_{n-1})$$

Для біта b_1 справедлива умова $b_1 = f(b_N)$, тобто

$$\forall i = 1, 2, 3 \dots \left[b_i = f(b_{(i-1) mod N}\right]$$

Тоді з урахуванням останнього співвідношення функціонування схем $SS_1 SS_2, SS_3$, керуючих деяким (будь-яким) і-м каналом формувача *(i = 1,2,3 ... N)*, можна описати наступними співвідношеннями:

Для схеми SS1:

$$SS_1\ D_i = a_i = b_{(i-1)modN} * g_i,$$

де: D_i- стан D-входу ЕП Rg T_i.

При цьому $(i = 1) \rightarrow (D_i = a_i = b_N \cdot g_i)$.

Для схеми SS_3:

$$C_i = g_i \cdot \tau,$$

де τ - вихід генератора синхросигналів зсуву.

Для схеми переноса SS_2:

$$b_i = b_{(i-1)modN * \overline{g_i} * g_i}$$

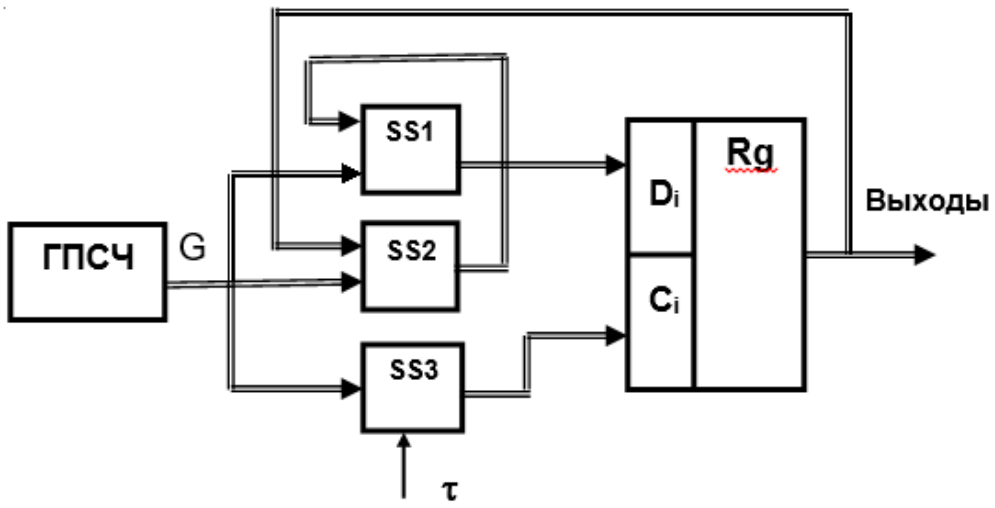

Рисунок 3.1 Узагальнена схема генератора псевдовипадкових двійкових послідовностей

На основі наведених логічних співвідношень функціональна схема каналу формувача $i = 1,2, \ldots N$ може бути представлена на рисунку 2.3, на якому з метою спрощення не показані ланцюги початкової установки ЕП регістра Rg.

Припустимо, що в процесі початкової установки регістра Rg в ньому з'явився набір, що містить *k* одиниць і *N-k* нулів (розподіл *k* одиниць по бітам регістра Rg не суттєво). Очевидно, що кільцеве з'єднання бітів регістра призведе до того, що вага будь-яких подальших наборів не зміниться і дорівнюватиме *k = 0,1,2, ... N-1, N*. Оцінимо ймовірність одиничного стану на довільному виході *i* регістра Rg (тотожність структури каналів і кільцеве з'єднання ланцюга перенесення дозволяє припустити, що наведені нижче співвідношення справедливі для будь-якого значення *i = 1,2, ... N*). Очевидно, що $P\ (T_i = 1) = P\ (a_i = 1)$, тобто ймовірність одиничного стану T_i дорівнює одиничному стану і-го виходу схеми А. Після досить великої кількості ПС-зрушень в середньому *k* одиниць будуть приблизно рівномірно розподілені по регістру Rg. Інакше кажучи, математичне очікування числа нулів між двома одиницями становитиме $N/k - 1$. Відповідно до рисунку нижче

$$P(a_i^1)=P(g_i=1)\cdot P(b_{(i-1)modN}=1),$$

де $P(a_i^1)$- ймовірність одиничного стану виходу a_i при $g_i=1$.

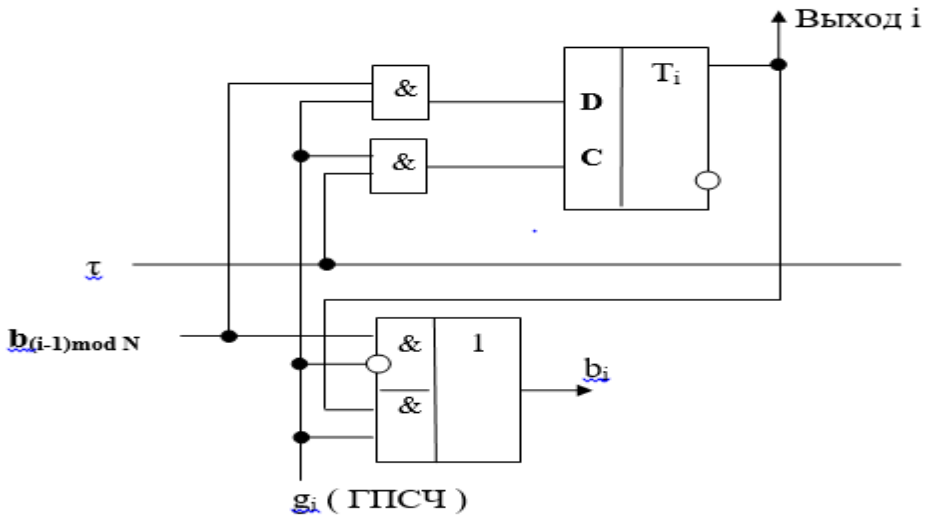

Рисунок 3.2 Функіональна схема формувача одиничного біта розробленого генератора псевдловипадкових двійкових біт

Аналогічно:

$$P(a_i^0) = P(g_i=0) \cdot P(T_i=1),$$

де: $P(a_i^0)$ – ймовірність збереження одиниці в ЭП T_i при $g_i=0$,тоді:

$$P(T_i) = \frac{1}{2} * \frac{1}{\frac{N}{k}} + \frac{P(T_i)}{2},$$

Отже,

$$P(T_i)=k/N$$

Тобто, змінюючи (задаючи) вагу вихідного коду в регістрі Rg в діапазоні $0 \div N$ отримаємо ряд можливих значень вихідних ймовірностей: *0, 1/N, 2/N, 3/N, ... (N-1)/N*, складність *L* такого формувача можна оцінити, виходячи з рисунку вище

$$L=2L(T)+5,$$

де *2L(T)*- сумарна складність реалізації ЕП *Ti* регістра *Rg* і розряду ГПВЧ.

Як показує аналіз структурних особливостей схеми SS_2, наявність послідовної ланцюга поширення сигналів *bi* призводить до лінійної залежності затримки в цій схемі від величини N розрядності вихідного набору формувача, і, як наслідок, до зниження тактової частоти роботи генератора, тобто до зниження продуктивності процедури отримання послідовності псевдовипадкових наборів заданої ваги, що і є основним недоліком даної схеми. Для збільшення швидкодії схеми допоможе додавання додаткової схеми групового переносу в регістрі зсуву генератора псевдовипадкових двійкових послідовностей.

4. Аналіз методів прискорення розповсюдження переносу

4.1 Використання регістру зсуву зі зворотнім зв’язком

У 1994 регістр зсуву зі зворотним зв'язком по перенесенню (FCSR) був винайдений Гореські і Клаппером, а також незалежно від них Марсаглією та Заманом. Не зважаючи на майже одночасне винайдення та схожість винаходу, вчкеними рухали різні цілі: Клаппер і Гореські хотіли використовувати його для криптоаналізу генератора, в той час як Марсаглія та Заман були націлені знайти хороший генератор випадкових чисел для вирішення таких завдань, як використання квазі-Монте-Карло методу

FCSR складається з зсувного регістру, функції зворотного зв'язку та регістру перенесення. Довжина зсувного регістру дорівнює кількості бітів. При потребі витягти біт, всі біти зсувного регістру зсуваються вправо на одну позицію.

Замість виконання операції XOR над усіма бітами відвідної послідовності ці біти складаються один з одним і з вмістом регістра перенесення. Результат mod2 і стає новим бітом. Результат, поділений на 2, стає новим вмістом регістра перенесення.

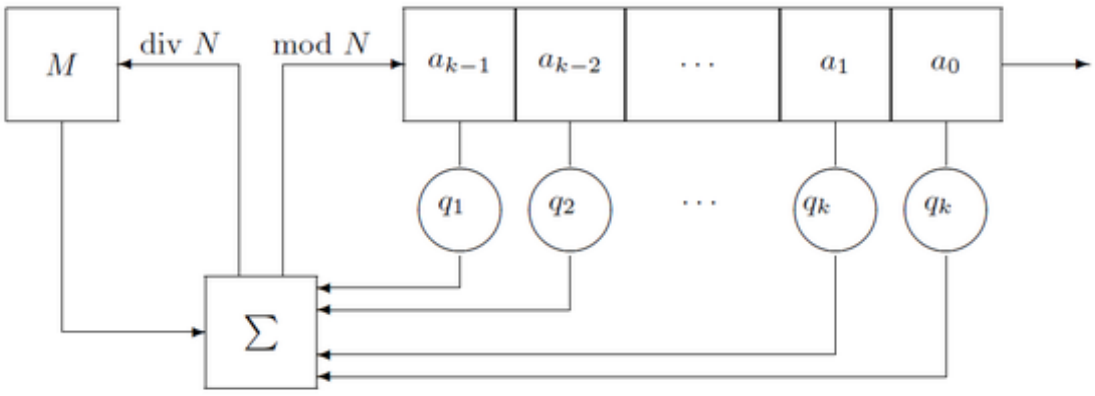

Рисунок 4.1 Регістр зсуву з оберненим зв’язком по переносу

Розглянемо приклад 3-бітового регістра з відгалуженнями в першій і другій позиціях. Нехай його початкове значення [0, 0, 1], а початковий вміст

регістра перенесення рівен [0]. Виходом буде крайній правий біт зсувного регістру. Подальші стани регістра наведено в таблиці нижче:

Номер шага	Сдвиговый регистр	Регистр переноса
0	0 0 0	0
1	1 0 0	0
2	0 1 0	0
3	1 0 1	0
4	1 1 0	0
5	1 1 1	0
6	0 1 1	1
7	1 0 1	1
8	0 1 0	1
9	0 0 1	1
10	0 0 0	1
11	1 0 0	0

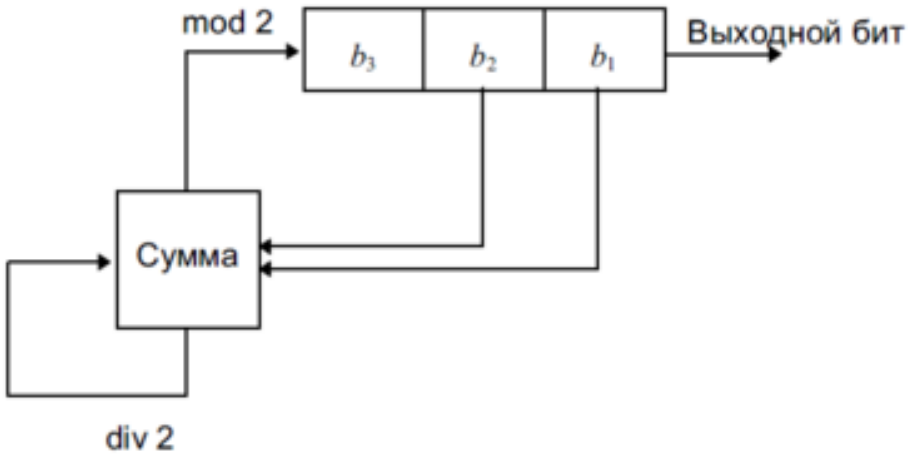

Рисунок 4.1 FCSR на три біта

Для FCSR існує затримка, перш ніж він перейде в циклічний режим, тобто почне генерувати циклічно повторювану послідовність.

Залежно від обраного початкового стану можливі 4 різних випадку:

1.Початковий стан може виявитися частиною максимального періоду.

2.Початковий стан може перейти в послідовність максимального періоду, після деякої початкової затримки.

3.Початковий стан може після початкової затримки породити послідовність нулів.

4.Початковий стан може після початкової затримки породити послідовність одиниць.

Дослідним шляхом можна перевірити, чим закінчиться конкретний початковий стан. Потрібно запустити FCSR на деякий час. (Якщо m - початковий обсяг пам'яті, t - кількість відгалужень, то досить $\log_2 m + \log_{g2} t + 1$ тактів.) Якщо вихідний потік вироджується в нескінченну послідовність нулів і одиниць за n біт, де n - довжина FCSR, то не варто використовувати цей початковий стан.

4.2 Варіанти реалізації регістру зсуву зі зворотнім зв'язком

Конфігурація Галуа

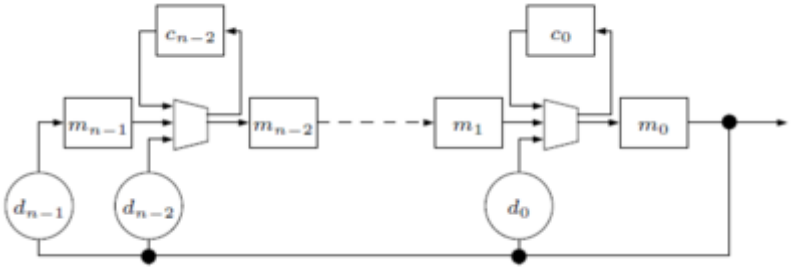

Рисунок 4.2 Кофігурація Галуа для FCSR

Конфігурація Галуа складається з:

n - бітного головного регістра $M = (m_0 \dots \ m_{n-1})$ з деякими фіксованими відгалуженнями $(d_1 \dots d_{n-1})$

n-1 - бітного регістра перенесення

Конфігурація Фібоначчі

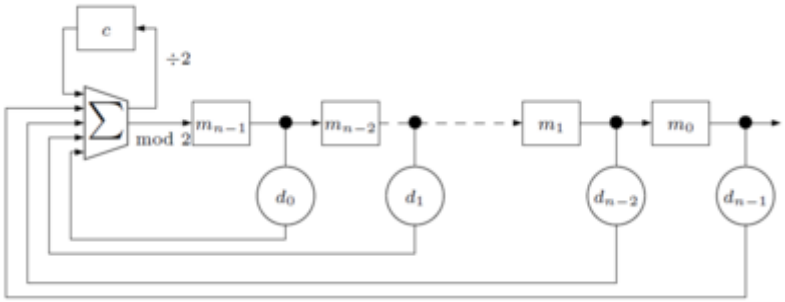

Рисунок 4.3 Конфігурація Фібоначчі для FCSR

Конфігурація Фібоначчі складається з:

n - бітного головного регістра $M = (m_0 \dots \ m_{n-1})$ з деякими фіксованими відгалуженнями $(d_1 \dots d_{n-1})$

Відгалуження $(d_1 \dots d_{n-1})$ пов'язані з регістром перенесення c, що складається з $w_h(d)$ бітів, де $w_h(d)$- вага Хаммінга для $d = (1 + |q|)/2$

4.3 Можливі варіанти використання в генераторах

Чергуючийся генератор “стоп-пішов”

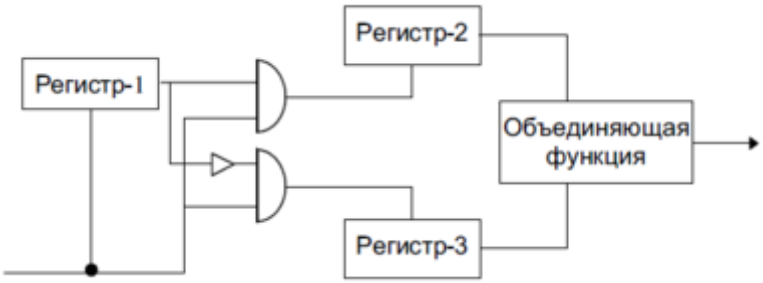

Рисунок 4.4 Чергуючийся генератор “стоп-пішов”

У ньому використовуються три регістри зсуву різної довжини.

Тут Регістр-1 керує тактовою частотою 2-го і 3-го регістрів, тобто Регістр-2 змінює свій стан, коли вихід Регістру-1 дорівнює одиниці, а Регістр-3 - коли вихід Регістра-1 дорівнює нулю.

Ці регістри використовують FCSR замість деяких LFSR, і операція XOR може бути замінена складанням з переносом.

Відомі способи використання регістрів зсуву в генераторах “стоп-пішов”:

1. Генератор «стоп-пішов» FCSR: Регістр-1, Регістр-2, Регістр-3 - FCSR. Об’єднуюча функція- XOR.
2. Генератор «стоп-пішов» FCSR / LFSR: Регістр-1 - FCSR; Регістр-2, регістр-3 - LFSR. Об’єднуюча функція - додавання з переносом.
3. Генератор «стоп-пішов» FCSR / LFSR: Регістр-1 - LFSR; Регістр-2, регістр-3 - FCSR. Об’єднуюча функція – XOR

Каскадні генератори

Дана схема являє собою поліпшену версію генератора «стоп-пішов». Він складається з послідовності регістрів, тактування кожного з яких керується попереднім регістром. Якщо виходом Регістру-1 в момент часу є 1, то тактується Регістр-2. Якщо виходом Регістру-2 в момент часу є 1, то

тактується Регістр-3, і так далі. Вихід останнього регістра є виходом генератора.

Існує два способи використовування FCSR в каскадних генераторах:

1. Каскад FCSR. Каскад Голлманна з FCSR замість LFSR.
2. Каскад FCSR / LFSR. Каскад Голлманна з генераторами, що змінюють LFSR на FCSR і навпаки.

Комбіновані генератори

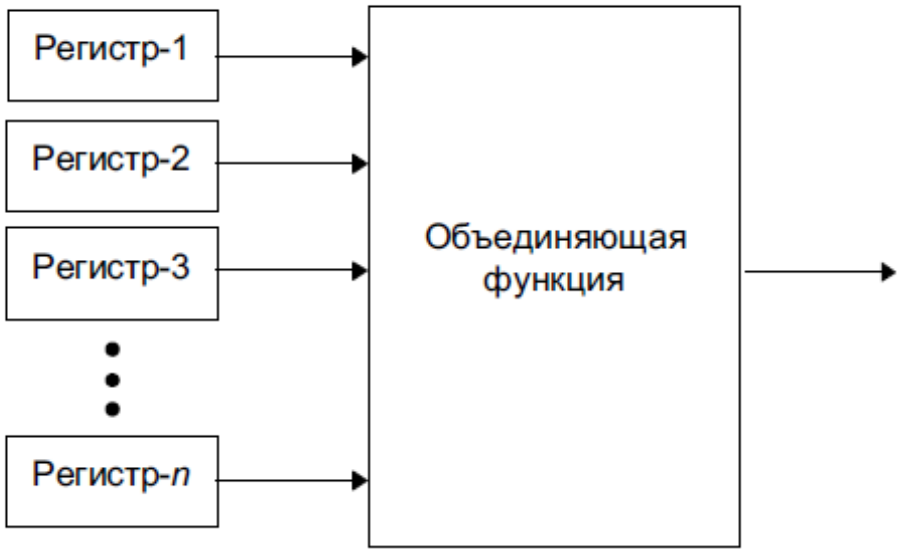

Рисунок 4.5 Комбінований генератор

Ці генератори використовують змінну кількість FCSR та LFSR і безліч функцій, які об'єднують регістри. Операція XOR руйнує алгебраїчні

властивості FCSR, тому має сенс використовувати цю операцію для їх об'єднання.

Відомі варіанти використання регістрів зсуву в комбінованих генераторах:

1. Генератор парності FCSR. Всі регістри - FCSR, Об’єднуюча функція - XOR.
2. Генератор парності LFSR / FCSR. Використовується поєднання LFSR і FCSR, Об’єднуюча функція - XOR.
3. Граничний генератор FCSR. Всі регістри - FCSR, а Об’єднуюча функція - мажорування.
4. Граничний генератор LFSR / FCSR. Використовується поєднання LFSR і FCSR, Об’єднуюча функція - мажорування.
5. Суммуючий генератор LFSR / FCSR. Використовується поєднання LFSR і FCSR. Об’єднуюча функція - додавання з переносом.
6. Суммуючий генератор FCSR. Всі регістри – FCSR. Об’єднуюча функція - додавання з переносом.

5.Використання групового переносу для пришвидшення роботи розробленого ГПВЧ

При роботі розробленої схеми генератора псевдовипадкових чисел найгіршим випадком буде ситуація, коли на виході генератора псевдовипадкових двійкових векторів будуть одні нулі. В такому разі ситуація до та після проходження синхросигналу зсуву не зміниться, отже час на проходження синхросигналу зсуву і виконання зсуву буде втрачено марно.

Можливим варіантом рішення цієї проблеми буде додавання схеми, призначення якої - відстеження ситуації ‘на всіх виходах нулі’ та заборона проходження синхросигналу зсуву до зміни цієї ситуації. Для генератора псевдовипадкових двійкових векторів з 4-ма виходами, схема зі зворотнім зв'язком по переносу буде виглядати так:

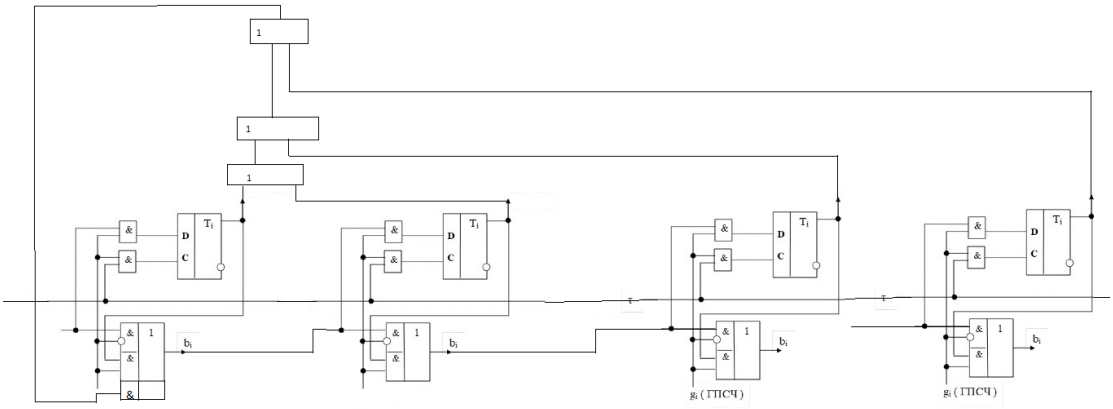

Рисунок 5.1 Схема генератора псевдовипадкових бінарних векторів на 4 біта з групуванням по 4.

Виходи i_1 та i_2 першого та другого елемента формування біта у схемі ГПВЧ з'єднані між собою через елемент "або". Вихід i_3 третього елемента з'єднаний з виходом першого елемента "або""(об'єднуючого i_1 та i_2).

Аналогічно, вихід i_4 об'єднано з виходом другого елемента "або" через елемент "або".

Вихід останнього елемента "або" надходить на третій додатковий вхід елемента "і" схеми формування синхросигналів. Таким чином синхросигнал зсуву пройде через групу елементів формування вихідої послідовності біт тільки в тому разі, якщо хоч один біт із сформованої вихідної послідовності дорівнює одиниці.

Для схеми на 4 біта рішення проблеми зайвих зсувів методом об'єднання усіх элементів в одну групу є логічним. Але при збільшенні кількості елементів шанс отримати комбінацію 'усі нулі' зменшується, а складність схеми об'єднання всіх елементів в одну групу зростає.

З іншого боку, при зростанні кількості елементів формування вихідних біт у схемі генератора псевдовипадкових двійкових векторів зростає ймовірність того, що на деякому значному відрізку двійкового вектора усі

елементи будуть рівні нулю. В такому разі не зважаючи на те, що отримана послідовність містить як нулі, так і одиниці, під час проходження синхросигналу зсуву через схему будуть зайві такти на відрізку з нулями. В такому випадку найдоцільнішим рішенням буле застосувати груповий перенос, тобто групувати не всі елементи в одну групу, як в попередньому прикладі, а розбити всі елементи на декілька груп, всередині яких забороняти проходження сигналу переносу в ситуації, коли всередині групи кожен елемент на виході дає нуль.

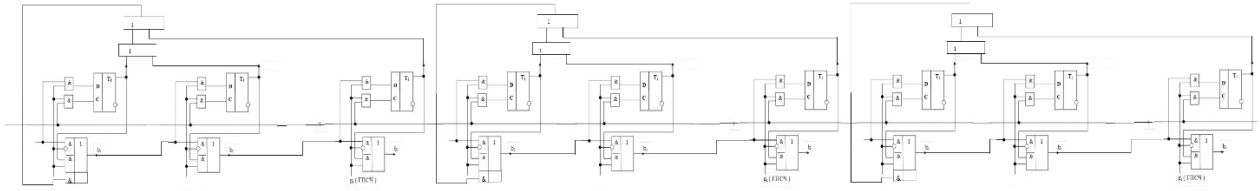

Рисунок 5.2 Схема розробленого генератора псевдовипадкових двійкових векторів на 9 біт з групуванням по 3 елемента

Наприклад, для генератора псевдовипадкових двійкових векторів на 9 розрядів доцільно буде застосувати групування по три елементи.

А в схемі на 8 біт доцільно буде застосувати групування по 4 елементи.

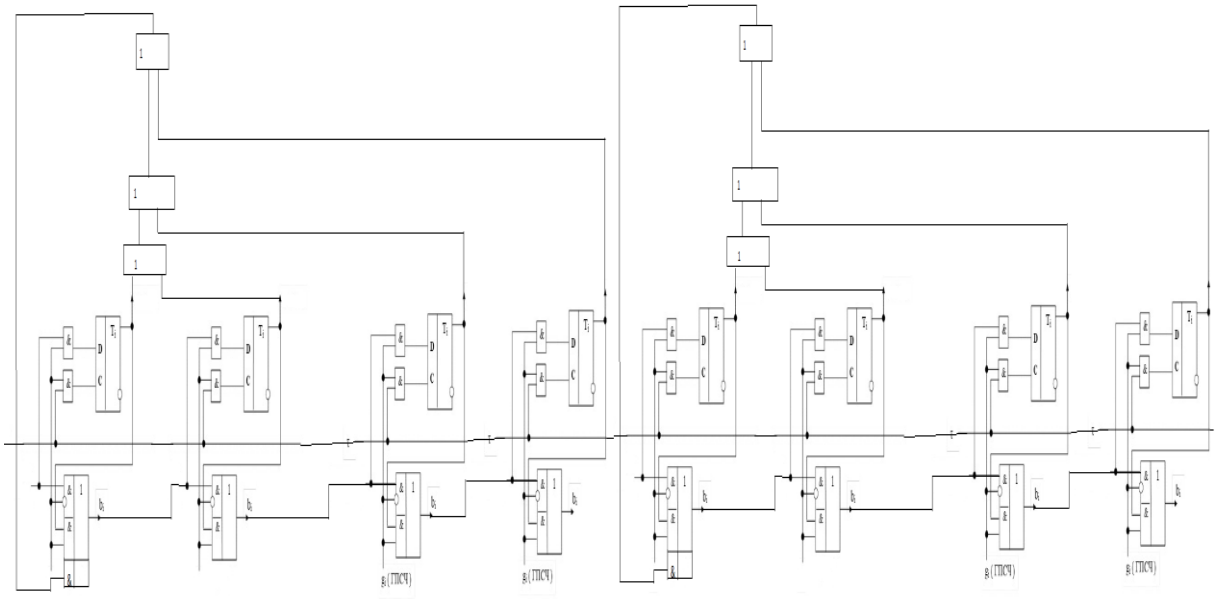

Рисунок 5.3 Схема розробленого генератора псевдовипадкових двійкових векторів на 16 біт з групуванням по 4 елемента

6.Математична модель залежності швидкодії схеми від кількості елементів схеми ГПВЧ з груповим переносом

Позначимо кількість біт в схемі генератора псевдовипадкових двійкових векторів як N, позначимо кількість груп для реалізації схеми групового переносу як g. В такому випадку, кількість елементів в одній групі:

$$n=N/g.$$

Також позначимо час затримки на схемі формування біта як t_b а час затримки на схемі групового переносу як t_p.

Розглядаючи роботу схеми більш детально, не складно зрозуміти, що при використанні схеми групового переносу можна виділити декілька основних станів при яких використовується схема групового переносу:

1. Всі нулі, груповий перенос задіяний у кожній групі.
2. В одній групі всі біти дорівнюють нулю.
3. В декількох, але не всіх групах всі біти дорівнюють нулю.
4. В кожній із груп є хоча б один ненульовий біт.

1. В такому разі загальний час роботи схеми буде дорівнювати:

$$T=g* t_p$$

2. В такому разі загальний час роботи схеми буде дорівнювати:

$$T= t_p+(N-1)* t_b$$

3. В такому разі загальний час роботи схеми буде дорівнювати:

$$T= t_p *n_g +(N-n*n_g)* tб,$$

Де n_g - кількість груп зі всіма елементами, рівними нулю

4. В такому разі загальний час роботи схеми буде дорівнювати:

$$T=g* t_b$$

Зрозуміло, що

$$t_b > t_p$$

Розглянемо також математичну модель залежності збільшення кількості елементів схеми від присутності або відсутності у схемі генератора псевдовипадкових двійкових векторів схеми групового переносу.

Позначимо кількість елементів в одиничній групі формувачів біт схеми генератора псевдовипадкових двійкових векторів як P_b , кількість елементів схеми групового переносу в одиничній групі формувачів біт -P_p .

У такому разі мінімально можлива для конкретного випадку кількість елементів, задіяних в схемі для основних станів, при яких використовується схема групового переносу буде дорівнювати:

1.Випадок “Усі біти дорівнюють нулю”:

$$P=g*(P_b + Pn)$$

2.Випадок “В одній групі всі біти дорівнюють нулю”:

$$P=g*P_b + P_p$$

3.Випадок “В декількох, але не всіх групах всі біти дорівнюють нулю”:

$$P= n_g \ *(P_b + P_p)+(g- n_g)* Pб$$

Де n_g - кількість груп зі всіма елементами, рівними нулю.

4. Випадок “В кожній із груп є хоча б один ненульовий біт.”:

$$P=g*P_b$$

Кількість елементів у схемі групового переносу для однієї групи можна вирахувати більш точно. У випадку запропонованої схеми групового переносу на кожні *n* біт генератора псевдовипадкових двійкових векторів приходиться *n-1* елементів “логічне АБО” та один додатковий вхід на схемі поширення

синхросигналу зсуву. Отже час затримки на схемі групового переносу буде дорівнювати:

$$t_p=(n-1)\,T_1 + T_{1зс}$$

де T_1-час затримки на елементі “логічне АБО”, а $T_{1зс}$-час затримки на додатковому вході схеми поширення синхросигналу зсуву.

У такому випадку час формули часу роботи схеми можна переписати у наступному вигляді:

1.Випадок “Усі біти дорівнюють нулю”:

$$T=g*((n-1)\,T_1 + T_{1зс})$$

2.Випадок “В одній группі всі елементи дорівнюють нулю”:

$$T=(n-1)\,T_1 + T_{1зс}+(N-n)*t_b$$

3.Випадок “В декількох, але не всіх групах всі біти дорівнюють нулю”:

$$T=((n-1)*T_1 + T_{1зс})*ng+(N-n*n_g)*t_b$$

Де n_g - кількість груп зі всіма елементами, рівними нулю

4.Випадок “В кожній із груп є хоча б один ненульовий біт.”:

$$T=g*t_b$$

Розглянемо також математичну модель збільшення складності схеми з додаванням у неї додаткової схеми групового переносу.

Як було зазначено в розділі 3 (Опис розробленого генератора псевдовипадкових двійкових векторів):

$$L=2L(T)+5,$$

де *2L(T)*- сумарна складність реалізації ЕП T_i регістра *Rg* і розряду ГПВЧ.

У випадку додавання схеми групового переносу:

$$L=2L(T)+5+L_n$$

Де L_n-загальна складність схеми групового переносу.

Виходячи з твердження, що складність реалізації схеми дорівнює кількості функціональних елементів схеми, та з формули часу затримки на схемі групового переносу $t_p=(n-1)*T_1+T_{1зс}$

$$L_n=n-1+1$$

$$L_n=n$$

Отже, складність реалізації однієї групи генератора псевдовипадкових двійкових векторів с додатковою схемою групового переносу:

$$L=n*(2L(T)+5)+n$$

А складність реалізації генератора псевдовипадкових двійкових векторів с додатковою схемою групового переносу:

$$L=g*(n*(2L(T)+5)+n)$$

7 .Опис програми, моделюючої ГПВЧ з груповим переносом

Для створення програми моделювання роботи створеної схеми було обрано мову програмування python3. Цю мову було обрано за наступними причинами:

1.Простота та читаємість коду на ній

2.Простота використання (і висока швидкість написання програм як наслідок)

3.Велика швидкість виконання програм

4.Існування великої кількості фреймворків для роботи з математичними моделями та графіками, що дозволяє пришвидшити роботу над моделюванням створеної схеми ГПВЧ.

Першим чином було написано програму для створення графіків зміни складності реалізації створеної схеми ГПВЧ із схемою групового переносу у залежності від кількості бітів вихідної послідовності генератора. Програма була написана з використанням фреймворків mathplotlib (для створення графіків) та numpy (для обробки математичних операцій).

Програма містить наступні змінні:

groups_amount-кількість груп для групового переносу, задається користувачем

bits_amount-кількість біт вихідної послідовності генератора, змінюється від числа, заданого користувачем, до числа заданого користувачем, з кроком, заданим користувачем

one_bit_genetator_complexity-складність реалізації схеми формування одиничного біту, може бути задана користувачем. В результаті аналізу схеми формування одиничного біту було вирішено, що стандартною величиною для one_bit_genetator_complexity буде 14

complexity-складність загальної схеми генератора псевдовипадкових чисел з додатковою схемою групового переносу, вираховується за формулою, описаною у попередньому пункті. На мові програмування python3 формула виглядає так:

complexity=groups_amount(bits_in_group*(one_bit_genetator_complexity+14)+ bits_in_group)*

При зміні кількості елементів від 0 до 1000 з кроком у 10, при групуванні елементів по 10, графік, згенерований програмою, виглядає так:

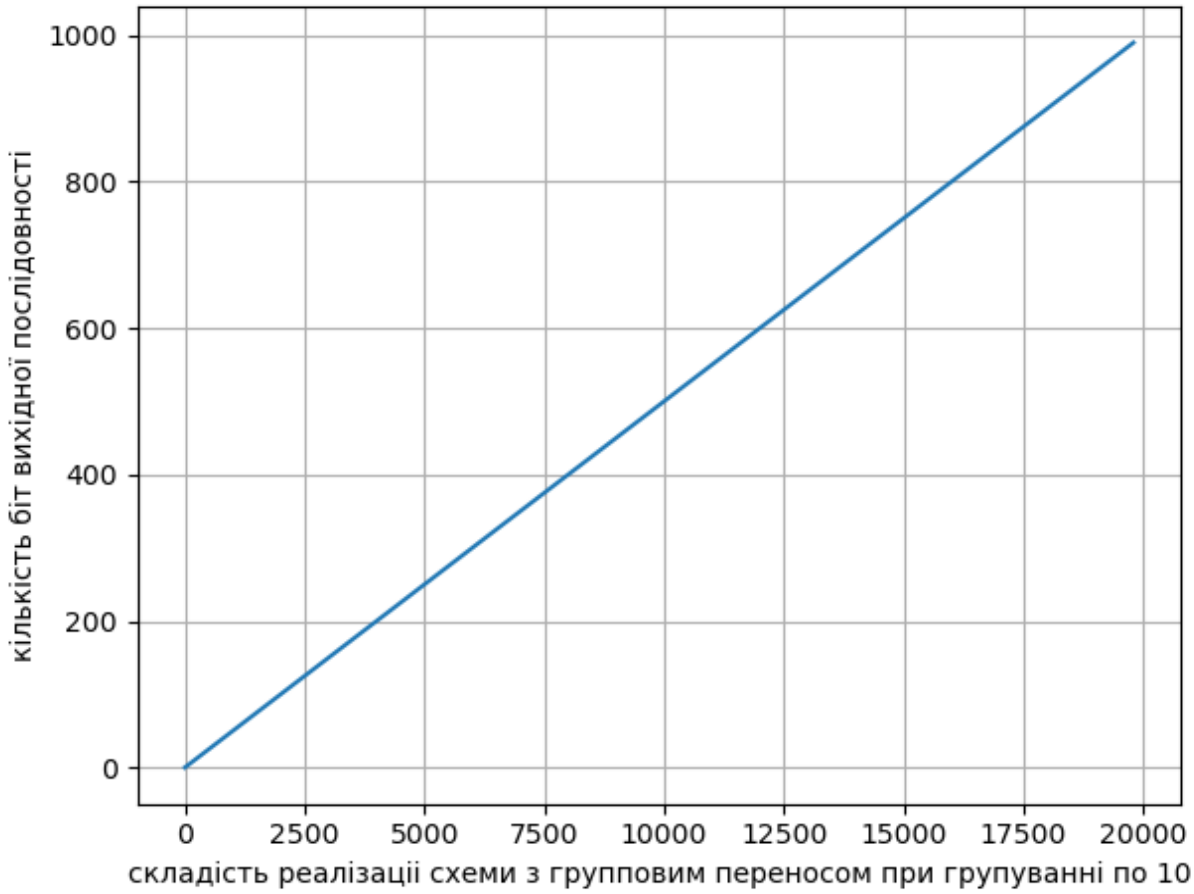

Рис7.1 Залежність складності реалізації схеми генератора с груповим переносом від кількості біт вихідної послідовності при групуванні по 10

Як видно на графіку, залежність складності реалізації схеми від кількості біт вихідної послідовності лінійна.

Дана ситуація залишається незмінною при будь-якому групуванні елементів.

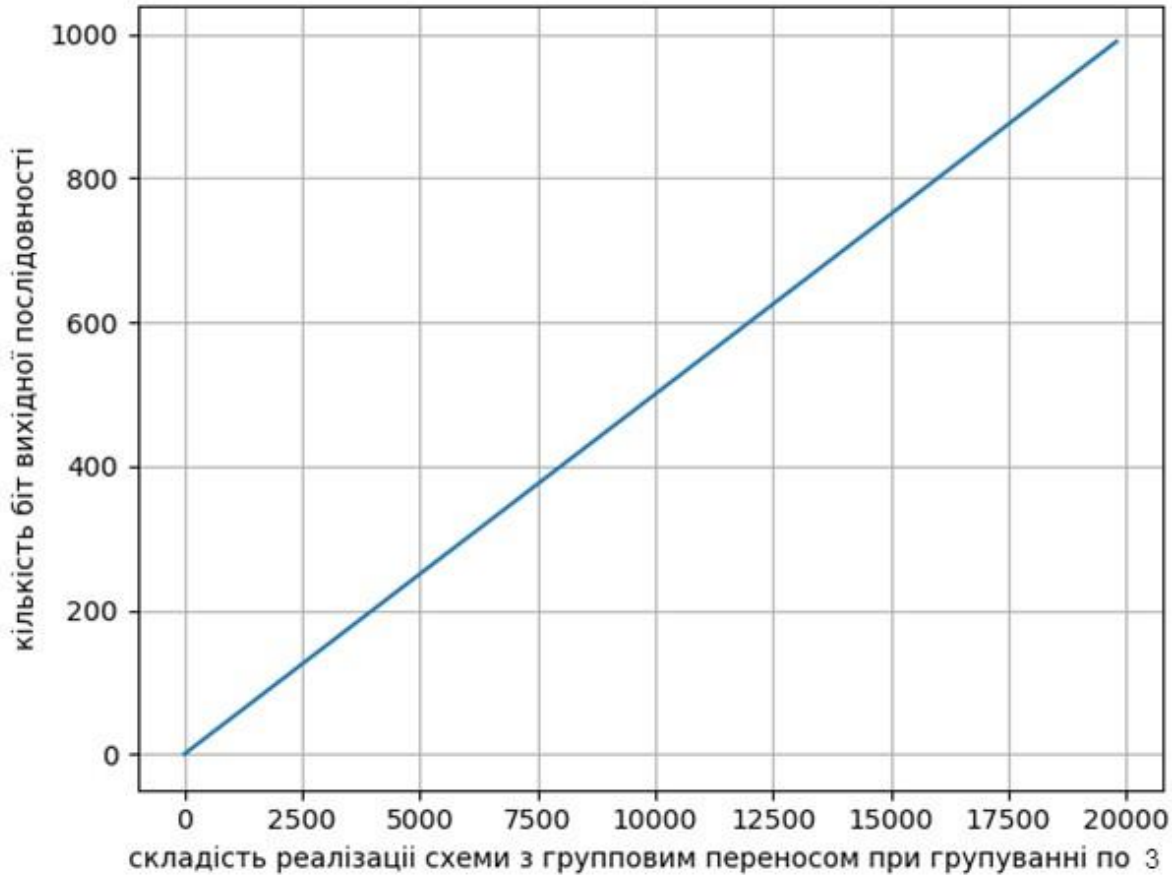

Рис7.1 Залежність складності реалізації схеми генератора с груповим переносом від кількості біт вихідної послідовності при групуванні по 3

Наступним кроком для моделювання створеної схеми генератора було написання програми для створення графіків залежності часу від кількості біт вихідної послідовності.

Програма містить наступні змінні:

groups_amount-кількість груп для групового переносу, задається користувачем

bits_amount-кількість біт вихідної послідовності генератора, змінюється від

числа, заданого користувачем, до числа заданого користувачем, з кроком, заданим користувачем.

one_bit_generator_time-час затримки на схемі формування одиничного вихідного біта, може бути заданий користувачем. В результаті аналізу схеми формування одиничного біту було вирішено, що стандартною величиною для one_bit_genetator_time буде 5.

group_shift_time-час затримки на схемі групового переносу, може бути заданий користувачем. У результаті аналізу схеми формування одиничного біту було вирішено, що стандартною величиною для group_shift_time буде 2.

null_groups_amount-кількість груп у яких задіяна схема группового переносу, дана величина отримана методом аналізу згенерованої вихідної послідовності генератора псевдовипадкових двійкових векторів.

binary_vector_result-змодельований програмно результат генератора псевдовипадкових чисел у вигляді бінарного вектора, також може бути заданий користувачем.

Після генерації бінарного вектора, програма аналізує його, підраховуючи кількість груп, в яких може бути застосований груповий перенос, та груп, у яких перенос неможливий, на основі цих даних програма визначає час роботи схеми для двох випадків:

1. Груповий перенос здійснено у всіх групах, в яких може бути здійснений груповий перенос
2. В жодній з груп груповий перенос не здійснювався

Звичайно, для детального аналізу зміни швидкодії схеми включення в неї додаткової схеми групового переносу недостатньо проаналізувати зміну швидкодії у випадку одного випадково згенерованого вектора сталої довжини.

Тому, наступним кроком написання програми було додавання функціоналу вирахування часу роботи схеми для двох випадків для певного діапазону розмірів вихідної послідовності генератора. У такому випадку робота з програмою виглядає так: користувач вводить мінімальну та максимальну розмірність вихідної послідовносте генератора, вагу вихідної послідовності генератора та кількість груп, на які відбувається розбиття. Після чого програма у циклі для кожної розмірності вихідної послідовності генерує випадковим чином (згідно с правилами роботи керованого генератора псевдовипадкових чисел, розробленого в цій роботі) бінарний вектор, аналізує його та вираховує час роботи схеми на цьому векторі для двох вищезазначених випадків.

Після цього розмір вихідної послідовності збільшується і процедура повторюється.

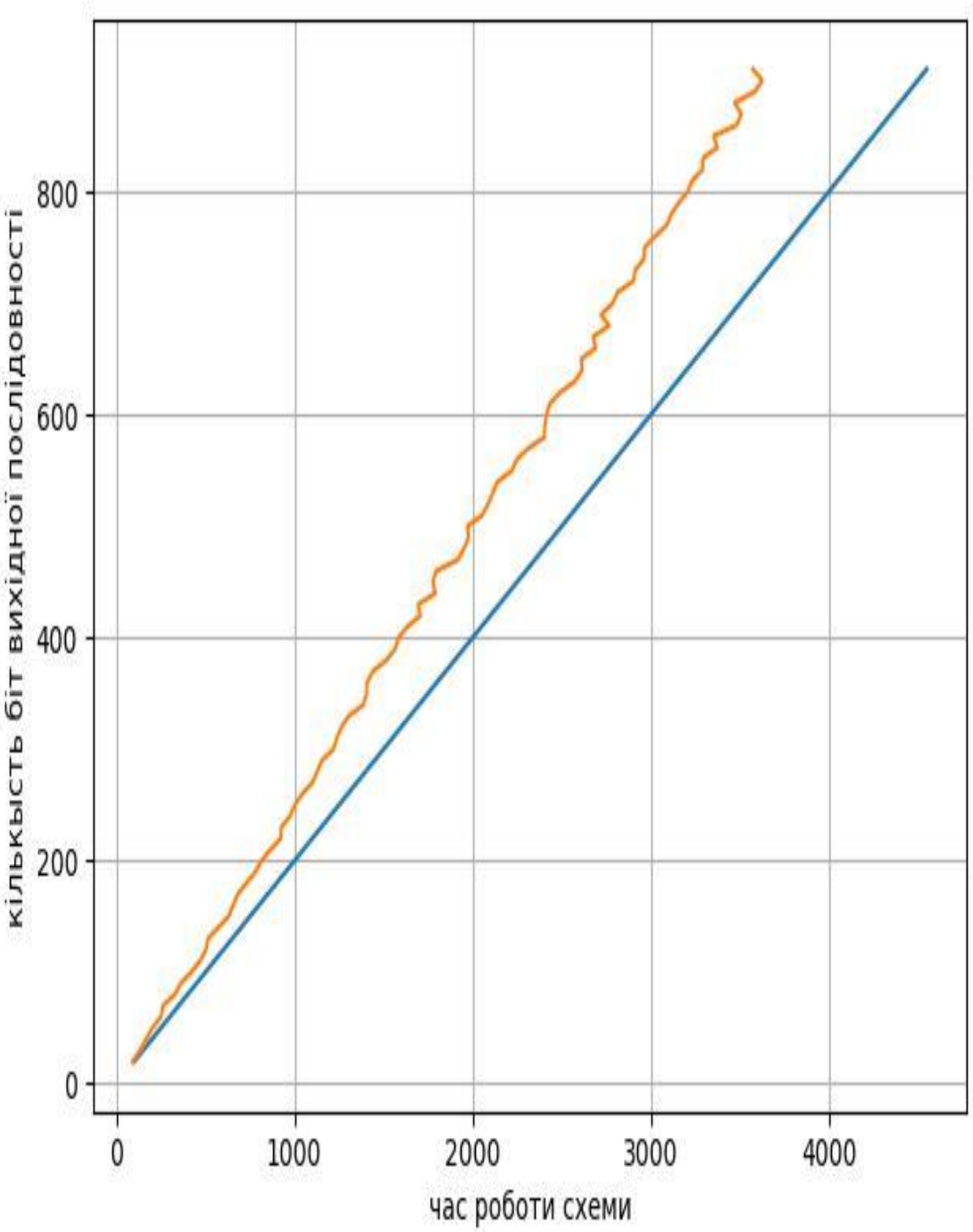

Рис7.3 Залежність часу роботи схем з груповим переносом та без групового переносу ід кількості біт вихідної посмлідовності

На рисунку вище показано результат роботи програми. По осі Y відкладено кількість біт вихідної послідовності. По осі X-час роботи програми.

Заміри проводилися для кількості біт від двох до 800, с кроком в 10 біт, вага вихідної послідовності була задана як ½ від кількості біт вихідної послідовності, кількість груп змінювалась з кроком в 5 груп а кількість біт в одній групі була сталою і дорівнювала 2.

Жовтим кольором на графіку показано залежність часу роботи схеми від кількості біт за умови включення у схему додаткової схеми групового переносу. Синім показано залежність часу роботи схеми від кількості біт за умови відсутності у схемі додаткової схеми групового переносу.

На графіку видно, що при заданих умовах схема з груповим переносом завжди виграє у часі схемі без групового переносу і розрив між ними збільшується зі збільшенням кількості біт вихідної послідовності генератора.

8.Результати тестування програми, моделюючої ГПВЧ з груповим переносом

Проаналізувавши результати роботи програми, моделюючої керований генератор псевдовипадкових двійкових векторів з додатковою схемою групового переносу, та без неї, можна стверджувати що в абсолютній більшості випадків включення у схему генератора додаткової схеми групового переносу дозволяє значно збільшити швидкодію схеми. З іншого боку додавання схеми групового переносу збільшить складність реалізації схеми. Як видно з наступних графіків, складність реалізації схем генератора з груповим переносом та без нього збільшується лінійно зі збільшенням кількості біт вихідної послідовності.

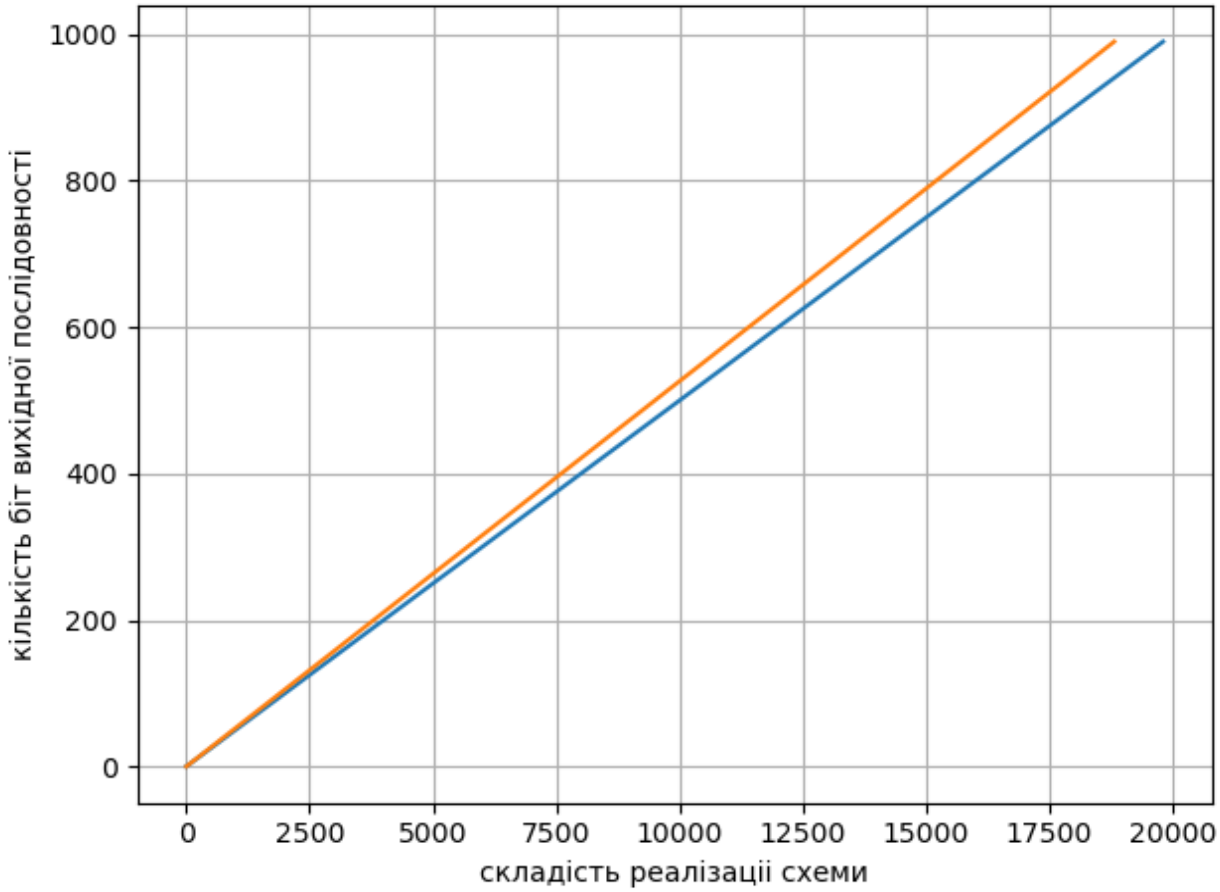

Рис 8.1 Залежність складності реалізації схем з груповим переносом та без групового переносу від кількості біт вихідної послідовності

На графіку вище жовтим кольором позначено залежність складності реалізації схеми генератора з груповим переносом від кількості елементів. Синім позначено залежність складності реалізації схеми генератора без групового переносу від кількості елементів.

Як видно, обидві залежності збільшуються лінійно, на відміну від залежності часу роботи схеми з груповим переносом, час роботи якої залежить від результату роботи генератора псевдовипадкових чисел.

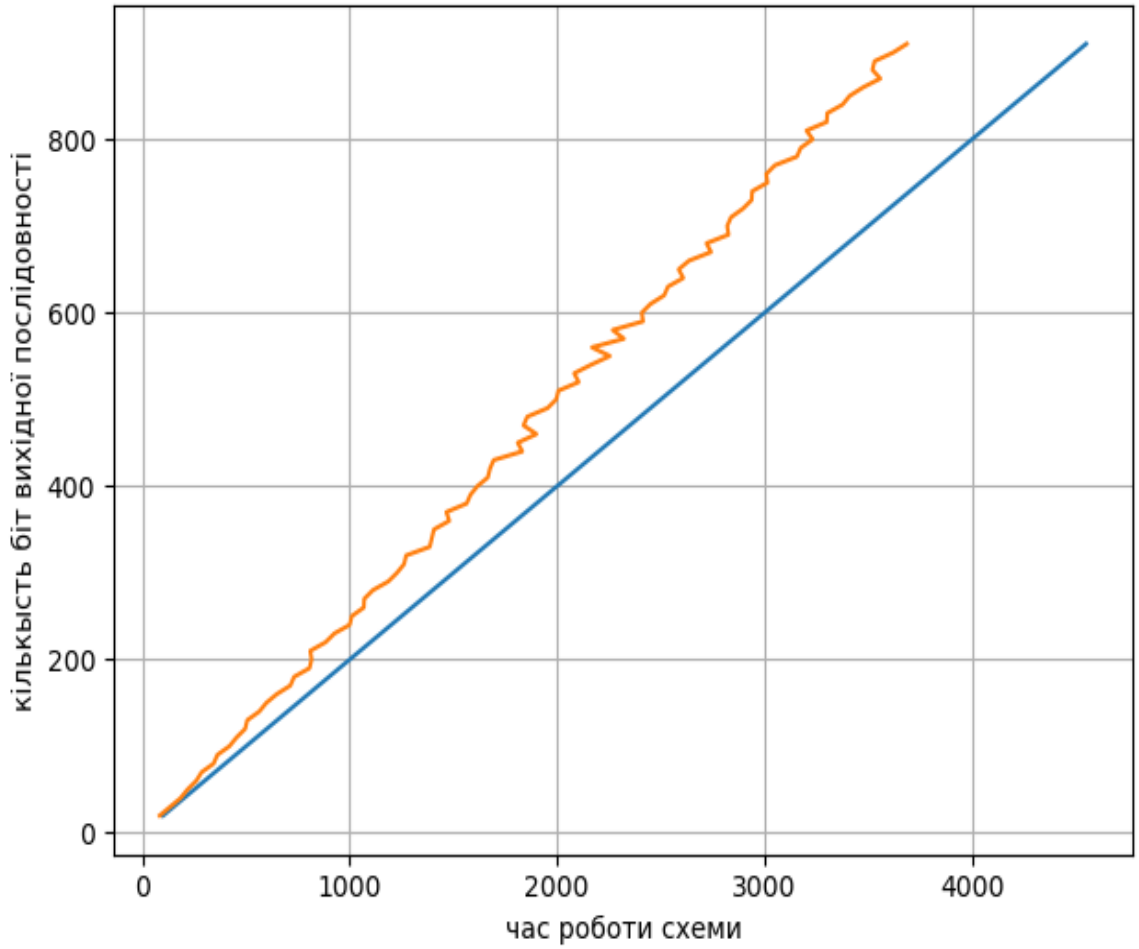

Рис 8.2 Залежність часу роботи схем з груповим переносом та без групового переносу ід кількості біт вихідної посмлідовності

Проаналізувавши графіки, можна зрозуміти, шо при збільшенні кількості біт вихідної послідовності, різниця між часом роботи схем збільшується на користь схеми з груповим переносом набагато швидше,

ніж збільшується різниця у складності реалізації схем на користь схеми без групового переносу.

Отже, виграш у часі роботи схеми з груповим переносом є значно більшим, ніж програш у складності реалізації даної схеми, і цей виграш збільшується зі збільшенням кількості елементів.

ВИСНОВОК

Були досліджені різні методи створення генераторів псевдовипадкових та випадкових чисел, після чого була розроблена власна схема керованого генератора псевдовипадкових двійкових послідовностей та створена програма, що моделює його роботу. Після аналізу результатів роботи та швидкодії цієї схеми було знайдено її слабкі місця та можливості для додаткової оптимізації роботи схеми.

Були досліджені різні методи оптимізації подібних схем та обрано метод додавання схеми групового переносу у схему керованого генератора псевдовипадкових двійкових послідовностей для підвищення швидкодії схеми.

Було побудовано математичну модель роботи схем генератора без додавання схеми групового переносу та з додаванням схеми групового переносу і знайдено закономірності зміни швидкодії обох схем залежно від кількості біт вихідної послідовності та зміни складності реалізації обох схем залежно від кількості біт вихідної послідовності.

Було доказано, що зі збільшенням кількості біт вихідної послідовності виграш у швидкодії схеми з груповим переносом зростає значно швидше, ніж програш у складності реалізації схеми з груповим переносом.

Було розроблено програму, що виводить графіки цих залежностей для ситуації, вказаної користувачем.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Слеповічев І.І ‘Генератори псевдовіпадкових чисел.’
2. https://studfile.net/preview/5367575/page:3/
3. https://www.qrz.ru/schemes/contribute/security/jammers/generator1.shtml
4. А. А. Кузнецов, А. С. Киян, Д. И. Прокопович-ткаченко, В. П. Зверев, Е. В.Котух, Т.Ю. Кузнецова ‘Доказуемо стойкий генератор псевдослучайных последовательностей для постквантового применения’
5. https://www.literaturki.net/elektronika/cifrovaya-shemotehnika1/166-summatory-s-gruppovoi-strukturoi
6. Амосов В.В ‘Схемотехника и средства проектирования цифровых устройств’
7. Н. Воробьев ‘Сумматоры: определения, классификация, уравнения, структуры и применение’
8. Гэри Гибсон ‘Tronix Book 2’
9. П. Хоровиц та У.Хилл «Мистецтво схемотехніки»
10. Амосов В. В. ‘Схемотехника и средства проектирования цифровых устройств’
11. https://ru.wikipedia.org/wiki/Регистр_сдвига_с_обратной_связью_по_переносу
12. В.А. Романкевич, И.В. Майданюк Структурный метод формирования двоичных псевдослучайных векторов заданного веса
13. В.В.Гроль Структурный метод генерации псевдослучайных последовательностей специального вида
- http://www.gelezo.com/radiofans_circuits/803000/803004/generator_sluchaiynih_chisel.html
14. https://www.intuit.ru/studies/courses/553/409/lecture/17868
15. https://ppt-online.org/97393

16..Подорожный, И. В. Обзор аппаратных генераторов случайных чисел .Курченко Андрей

17.Реализация генератора случайных чисел на микроконтроллере MSP43